49th IEEE Conference on Decision and Control
December 15-17, 2010
Hilton Atlanta Hotel, Atlanta, GA, USA

Secure Network-Wide Clock Synchronization in Wireless Sensor Networks

Roberto Solis Robles Jason J. Haas Jerry T. Chiang Yih-Chun Hu P. R. Kumar
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
{robsolis, jjhaas2, chiang2, yihchun, prkumar}@illinois.edu

Abstract— **Clock synchronization is of critical importance for several applications in wireless and mobile sensor networks; for example to determine the order and time of events. Although several protocols to achieve clock synchronization have been developed, they may not be secure. For instance, a link controlled by an attacker could delay packets it forwards in ways which would cause the nodes sharing that link to obtain faulty time estimates. In this paper, we propose a secure network-wide clock synchronization protocol. It also allows nodes to securely discover the network topology by detecting and isolating links that behave inconsistently. This network-wide clock synchronization protocol is built on ideas in [1] where inconsistent attacks are detected using timing information alone under certain conditions. The proposed protocol has been implemented and the results of experimentation on a twenty-five node network are presented.**

I. Introduction

Wireless sensor networks have been extensively used in designing novel wireless and mobile applications that range from environmental monitoring and object tracking, to control over networks. Many of these applications require the entire network to have a common notion of time in order to function or to operate optimally. We present a protocol to provide a secure network-wide clock synchronization.

While many of the previously proposed synchronization protocols provide a certain level of Byzantine fault tolerance, they are susceptible to wormhole attackers who simply route messages via a different path rather than altering the integrity of the message [2]. A wormhole attacker can change the measured round trip time and other link attributes, for example by delaying a packet during forwarding. Therefore, synchronization protocols, under wormhole attacks, can result in clocks that are in reality not synchronized at all. In this paper, we broadly call all attacks that result in non-constant link delays or skews as *inconsistency attacks*, whether these attacks originate in the link or at one of the end points. We call an attack an *inconsistency based attacker* if it performs such inconsistency attacks.

Recently, Chiang et al. [1] used the algorithm presented by Graham and Kumar [3] and Freris and Kumar [4], and proposed a protocol that can detect all half-duplex and some full-duplex inconsistent attackers by using timing information alone. Assuming that all clocks in the network are affine, Chiang et al. showed that a half-duplex attacker will be forced into the position of having to send and receive packets acausally in order to remain undetected. They also showed that even a full-duplex attacker located sufficiently far away from both end points of a link needs to have the technologically hard capability to send and receive two messages simultaneously in order to avoid being detected. In other words, a single full-duplex attacker in certain locations, and a half-duplex attacker, will always be detected by the protocol proposed by Chiang et al..

In this paper, we extend the work done in [1] from a single-link protocol to a secure global network synchronization protocol. We globally synchronize by examining every path possibly used by half- and full-duplex attackers. We then remove any links on which a misbehavior is detected. Since we need to examine all paths to remove misbehaving links, our protocol also allows a node to learn and construct the topology of the network.

To the best of our knowledge, our protocol is the first to achieve global clock synchronization with the following properties:

1) It ensures that when one node misbehaves and a link is subsequently removed from the network, the eliminated link must be incident on the misbehaving node.
2) Our protocol gives individual nodes sufficient information to attribute detected misbehavior; that is, a node does not require cooperation of peers in order to ignore misbehaving links.
3) Finally, our protocol provides several capabilities which follow from knowledge of topology, such as secure source routing and secure link state routing.

We have implemented the proposed protocol on Crossbow IMote2 hardware [5], which uses the TinyOS [6] operating system. Our implementation is, to our knowledge, the first implementation of a clock synchronization protocol that allows a node to securely discover the network topology by detecting and isolating all links that do not comply. The remainder of this paper is organized as follows. In Section II, we discuss background material including an introduction on wormhole attacks and related work on clock synchronization and security. In Section III, we present our main contribution,

This material is based upon work partially supported by USARO under Contract No. W-911-NF-0710287, AFOSR under Contract FA9550-09-0121, and NSF under Contract Nos. CNS-05-19535, CNS-07-21992 and CNS-0626584.

the protocol. We describe the implementation in Section IV. In Section V, we present the main results of the experimental evaluation. Finally, we conclude the paper in Section VI.

II. Background

We now define terms that are used in the remainder of this paper, and present previous work on clock synchronization. In particular, we present the results of [3], [4] and [1], which lay the foundation for our network-wide clock synchronization protocol.

A. Wormhole Attack

In this paper, we consider a wormhole attacker to be an attacker that does not seek to alter messages, but rather seeks to alter message delay (i.e., the time from sending to receiving). We define *consistent* packets as packets that an attacker delays by a constant amount, and we call such an attacker a *consistent* attacker. If an attacker cannot or does not inject consistent delay, then we call such an attacker an inconsistent attacker. We call skews measured on one particular path *consistent* if they are approximately the same over time. We call skews measured from different paths *compliant* if they are approximately the same over different paths.

Wormhole attackers, consistent or inconsistent, can be classified into three groups according to their communication abilities:

1) A *half-duplex* attacker can only receive or transmit one message at any instant in time.
2) A *full-duplex* attacker can only receive and transmit one message at any instant in time.
3) A *double-full-duplex* attacker can receive two messages and transmit two messages at the same time.

B. Clock Synchronization

Clock synchronization has been extensively studied for distributed systems over both traditional networks and over sensor networks. Graham and Kumar [3] provided fundamental limits of timestamp-based clock synchronization with affine clocks in a trusted environment. Freris and Kumar extended the work to networks. They showed the feasibility of a node determining precisely its clock offset and skew with respect to a global reference clock [4]. In particular, they showed that after clock synchronization, the transmitting node can perfectly predict the time of the receiving node at which a transmitted packet will arrive.

Several protocols have been proposed for clock synchronization. Elson et al. [7] proposed the Reference Broadcast Synchronization (RBS) algorithm, where a node transmits a reference packet to its neighbors. Ganeriwal et al. [11] and Maróti et al. [12] also present algorithms to synchronize a node with its neighbors. Solis et al. [8] presented and implemented a completely asynchronous and distributed algorithm that makes use of several global constraints that need to be satisfied by a common notion of time in a multi-hop network. Giridhar and Kumar [9] analyzed this algorithm, building on a similar technique as used in Karp et al. [10], and showed that under a *stochastic model* of offsets in a spatial network, the synchronization error is $O(1)$ as the number of nodes in a random connected graph increases, thus providing support for the feasibility of accurate synchronization in large *wireless networks*.

Using time-diffusion synchronization, Su and Akyildiz [13] present protocols that can be used to synchronize an entire network.

Li and Rus [14] proposed several protocols for global clock synchronization: an all-node based, a cluster based, and two diffusion based algorithms.

Regarding security, there are studies [15], [16] that describe the possible attacks that could be performed on clock synchronization protocols such as RBS [7], TPSN [11] and FTSP [12], but very few clock synchronization protocols have been proposed to handle these attacks.

Ganeriwal et al. [17], proposed secure protocols to handle a certain type of attack the authors call *pulse-delay attack*, where an attacker delays the delivery of a synchronization packet, which results in an incorrect calculation of timing parameters at the receiver. The mechanism used to authenticate the transmitted messages only works for sensors with low data rates, and to synchronize a group of nodes they all need to be within range, making it unsuitable for multihop networks.

Sun et al. [18] devised protocols for securely synchronizing clocks of sensor network nodes to global time servers. They develop two protocols that are able to synchronize node clocks in a fault-tolerant manner. They do not describe how neighbors are discovered, nor do they consider the security properties of their protocol if neighbor discovery is subject to attack.

Chiang et al. [1] extended the clock synchronization protocol of [3] and [4] to detect inconsistent attackers. If a wormhole attacker inconsistently delays packets over a link, the two nodes would measure the other's skew as being time-varying, thereby detecting the attacker. In other words, a wormhole attacker must inject consistent delays in order to avoid being detected. Besides showing that a wormhole attacker must be consistent in order to avoid being detected, [1] also showed that half-duplex and sufficiently distant full-duplex wormhole attackers will also eventually be detected regardless of their consistency.

C. Secure Clock Synchronization Between Two Neighboring Nodes

We now describe in detail the ideas presented in [3] and [4] and extended in [1] to provide security against wormhole attackers. We assume the nodes in the network use affine clocks. That is, the local time at node A has the value

$$A(t) = S_A t + o_A,$$

where t is the time at some global reference, $S_A > 0$ denotes the *clock skew*, which is the relative speed of node A with respect to the global reference, and o_A denotes the *offset* with respect to the global reference.

Graham and Kumar [3] showed that exchanging timestamps cannot yield a synchronization protocol that allows two affine clocks to agree on time with perfect precision. Although two clocks cannot completely agree with each other, Freris and Kumar [4] showed that it is feasible for the sender to perfectly *predict* the time on the receiver's clock at which the packet would arrive. For example, a basic clock synchronization for two nodes A and B can be achieved by exchanging four packets as shown in Figure 1. $P_{AB}(n)$ denotes the n^{th} packet sent from node A to B. This packet includes the time at which it is sent, $\sigma_{AB}(n)$, and the last time at which a packet was received in the reverse path, $\rho_{AB}(n-1)$. After these four packets have been exchanged, node A has enough information to calculate an estimate of the relative skew of B's clock with respect to A, $\hat{S}_{AB}$, and an estimate of the one-way delay from A to B, $\hat{\delta}_{AB}$, as follows:

$$\hat{S}_{AB} = \frac{\rho_{AB}(n) - \rho_{AB}(n-1)}{\sigma_{AB}(n) - \sigma_{AB}(n-1)}, \qquad \hat{\delta}_{AB} = \rho_{AB}(n) - \hat{S}_{AB}\sigma_{AB}(n).$$

One more packet from A to B, $P_{AB}(n+2)$, would allow node B to perform a similar calculation for the reverse link. After performing this basic clock synchronization, legitimate nodes can exactly predict the time at which the packet $P(n+1)$ will be received in the receiving node's clock by using the following:

$$\hat{\rho}_{AB}(n+1) = \hat{S}_{AB}\sigma_{AB}(n+1) + \hat{\delta}_{AB}.$$

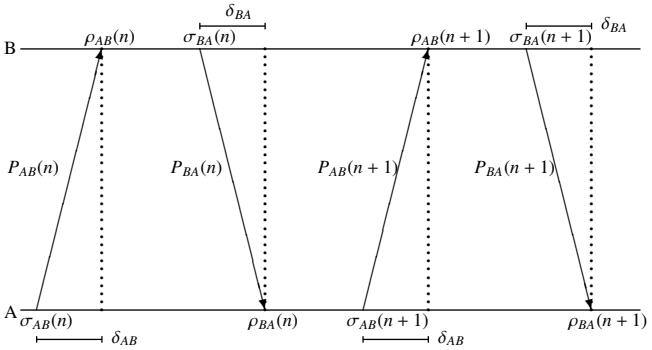


Fig. 1. Basic clock synchronization

References [3], [4], [1] collectively make the following assumptions in their protocols.

1) Nodes can accurately timestamp packets.
2) Nodes that share a link trust each other.
3) Nodes sharing a link also share a secret key, which is used to provide authentication and confidentiality. Using the private keys in this way eliminates traditional, cryptographic man-in-the-middle attacks.
4) Attackers are unable to break the cryptography used in this protocol.
5) The product of the true skew of one node relative to another node and the skew of the reverse pair must equal to 1. That is, $S_{AB}S_{BA} \approx 1$ for all node pairs $\{A, B\}$.
6) An attacker cannot violate causality to produce negative delays.

III. Secure Network-wide Clock Synchronization Protocol

We extend [3], [4], [1] to a multi-hop network so that links on which wormhole attackers are detected are ignored for the purposes of routing. That is, our protocol will securely synchronize the clocks used in a multi-hop network.

TABLE I
Notation

Symbol	Definition (units)
τ_A	Turn-around time of node A (s)
r	Round-trip time (s)
r'	r + delay by attacker (s)
m_{AB_i}	Attacker delay of packet i from A to B (s)
o_A	Clock offset of node A with respect to the global reference (s)
S_A	Clock skew of node A with respect to the global reference
S_{AB}	Relative clock skew of node B to node A, $S_{AB} = S_B/S_A$
δ_{AB}	One-way delay from A to B
N_{AB}	The total number of packets sent so far from A to B
$\rho_{AB}(n)$	B's local clock time when the n^{th} packet from A was received
$\sigma_{AB}(n)$	A's local clock time when it sent its n^{th} packet to B
$P_{AB}(n)$	n^{th} packet sent from A to B

To describe our protocol, we use the same assumptions used by [1] as stated in Section II-C and use the notation summarized in Table I. The ultimate goal of our protocol is to obtain a network-wide *compliant* clock, that is, for any two nodes A and B in the network, the skews of A measured by B through k different paths between them should be approximately equal. If the skews obtained do not have this property, then we say the skews are *non-compliant* (e.g., one or more links have been compromised), and we remove the incorrect ones.

Our protocol has the following properties:

1) **Detection.** The protocol is able to detect non-compliant links and tag them.
2) **Dissemination.** Once a node detects a non-compliant link, an "alert" is disseminated throughout the network.
3) **Isolation.** When a node A wants to communicate with a node B and there are non-compliant links along one or several paths between them, node A will verify which path(s) contain these non-compliant links and will remove them from its internal list, effectively isolating them.

Our protocol seeks to provide a network-wide compliant clock and requires each node to trust its own clock. It can be divided into four steps: single link check, neighborhood check, network compliance check, and removal of network non-compliance.

a) Single Link Check: The protocol starts by having every node in the network synchronize with each of its neighbors using the secure clock synchronization protocol proposed in [1]. Two nodes A and B sharing a link can collaborate and use timing information alone to measure and exchange the estimated relative skews $\hat{S}_{AB}$ and $\hat{S}_{BA}$. Reference [1] showed that this step allows nodes to detect any links controlled by inconsistent attackers by testing whether $\hat{S}_{AB}\hat{S}_{BA} \approx 1$. Each node also uses the measured skews to constantly monitor the consistency of its links. In

our evaluation, we show that this test is usable in networks comprised of motes.

b) Neighborhood Check: In the next step, node A then compiles a neighbor list, excluding any neighbor M where $\hat{S}_{AM}\hat{S}_{MA} \not\approx 1$ (such a link is deemed to be controlled by inconsistent attackers). Neighbors on this list are considered as *valid neighbors*. Node A also records the measured skew and round trip time of each valid neighbor. Node A then distributes the valid neighbor list to each valid neighbor. By eliminating all inconsistent links, a node is assured that all links incident on itself are either direct or controlled by consistent attackers. Then for every consistent link incident on each node, the node records all its valid neighbors into its *list of reachable nodes* (we will distinguish these two lists below) by recording the ID's, measured skews, and measured round trip times of the neighbors that can be reached in one-hop. Each node then distributes its list of reachable nodes (right now simply a list of valid neighbors) to its valid neighbors.

c) Network Compliance Check: Every time a node receives a list of reachable nodes from its neighbor, the node adds any new nodes from loop-free paths to its list of reachable nodes (the list initially only contains its valid neighbors). Updating the list of reachable nodes involves calculating the aggregate skew and aggregate round trip time. In particular, a node needs to verify the compliance of the end-to-end relative skews, which is calculated by taking the product of all relative skews on a path. That is, if node A knows that it can reach B through X or Y, then A must examine whether $\hat{S}_{AB,P_1}$ is equal to $\hat{S}_{AB,P_2}$, by comparing $\hat{S}_{AB,P_1} = \hat{S}_{AX}\hat{S}_{XB}$ and $\hat{S}_{AB,P_2} = \hat{S}_{AY}\hat{S}_{YB}$. The skew of node B measured by node A is *valid* if the inverse relationship holds when A and B share a link, or if the aggregate skews from different paths yield compliant skews, otherwise, node B would be recorded as an *invalid* reachable node.

Every time a node makes additions to its list of reachable nodes, the node floods the updated reachable-nodes list to its neighbors. The flooding will terminate in finite time when every node in the network finds out about the network topology. We note that the procedure used in this step is similar to the one performed by the distributed Bellman-Ford algorithm [19], but we are aggregating the skew and round-trip time, instead of computing a minimum cost path.

d) Removal of Network Non-compliance: If non-compliant skews are detected between two nodes, at least one aggregated skew is different from the true skew between the two nodes. A node can detect the misbehaving path by sending a packet to the other node and specifying the arrival time. Since the misbehaving path exhibits incorrect skew, a misbehaving entity (be it a node or an attacker that controls a link) must be inconsistently delaying the packets because consistent delays do not alter the measured skew. Thus, similar to the results shown in [1], such a non-compliant path must eventually *acausally* delay packets to maintain its fast skew or result in a time-out to maintain its slow skew. Both cases will result in the removal of the non-compliant path.

If we do not require network-wide synchronization, but only an on-demand node-to-node synchronization, then we can eliminate the non-compliance in only the paths connecting the nodes of interest.

IV. Implementation Details

To verify the properties of our protocol, we implemented it as described in Section III on a testbed built using Crossbow's IMote2 [5] Zigbee sensor nodes using the TinyOS 2.1 operating system [6]. Our implementation consists of the following major parts. First, we use exponential smoothing to handle quantization errors in timestamps and the drifts in clock skew over time. Second, we generate accurate timestamps at the MAC layer in a fashion to similar previous work [7], [11], [12], [8]. These timestamps are required to verify that the use of affine clocks is a sound assumption, and to predict the remote node's clock time at which a packet will arrive. Third, we use a neighbor discovery protocol that permits a node to detect replayed packets. Fourth, we exchange neighbor lists and reachable lists between nodes with enough information to allows the nodes to incrementally discover new nodes in the network, learn new paths to previously known nodes, and verify the compliance of skews along the different paths as described in Section III. The reachable lists are exchanged using a timer mechanism to avoid a heavy traffic load. Finally, each node removes inconsistencies that it discovers.

V. Evaluation

In this section, we first show that over a single link, two nodes can perform clock synchronization such that a sender can predict a packet's arrival time according to the receiver's clock accurate to a few microseconds. We then show that our implementation of our network-wide secure clock synchronization protocol has the following property: whenever a legitimate node tries to use a misbehaving link, the legitimate node is able to detect misbehavior after a bounded amount of time and isolate the misbehaving link.

A. Clock Synchronization Between Two Neighboring Nodes

The clock synchronization behavior depends on two assumptions: that the product of actual skews is approximately one, i.e., $S_{AB}S_{BA} \approx 1$, even when independently measured, and further that our timestamps are sufficiently accurate to detect small perturbations in the RTT. To validate these assumptions, we ran the first step of our protocol with two motes and no attackers. We collected samples from over 1600 runs of the synchronization protocol. Each sample corresponds to one exchange of packets: one packet from A to B and one packet from B to A. We examined the skews, the product of the skews, the change in estimated offset, and how this change impacted the accuracy of our ETA (Estimated Time of Arrival).

The measured skews and their products are shown in Figure 2. All three measurements are close to 1, thus, we subtracted 1 from all data points in order to display the scale properly. Even though the skews are measured independently,

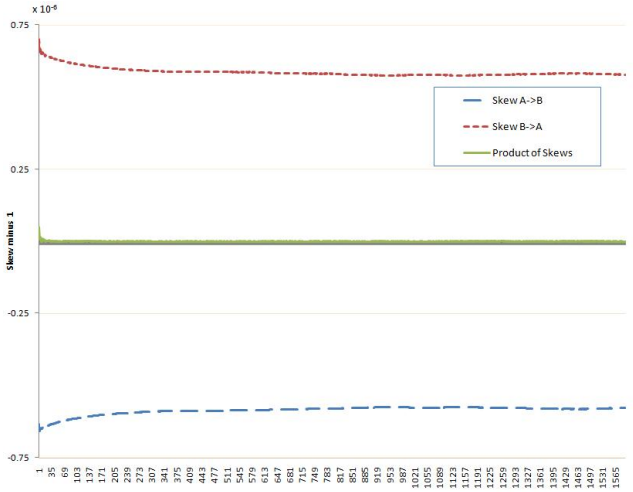


Fig. 2. Measured skew

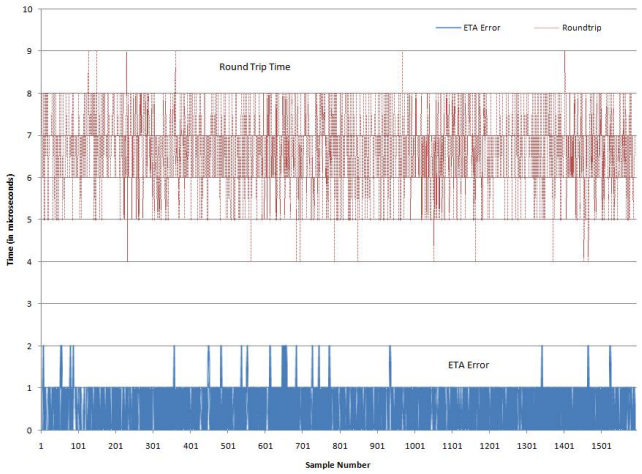


Fig. 3. Accuracy of packet arrival time prediction

the product of the skews is always very close to 1, thus validating the first assumption stated in this section.

Figure 3 shows the accuracy of our ETA. The y-axis is shown in microseconds. The median error is 0 μs and the average error is 0.5 μs, which reflects a high level of accuracy since the clocks we used only had 1 μs granularity. This result shows that nodes are able to accurately predict the time at which a receiver will receive a packet according to the receiver's clock, again validating our assumption.

B. Detecting Consistently Modified Skew in Multi-hop Network

We showed in Section III that a malicious proxy who seeks to consistently modify the measured aggregated clock skew between two nodes will eventually have to acausally delay packets in one direction and thus cannot remain undetected. In this section, we discuss our implementation of the detection algorithm. In our experimental setup, there are 5 nodes, N_1 through N_5 as shown in Figure 4(a).

Suppose N_1 and N_5 measure non-compliant skews through the two paths, $N_1 - N_2 - N_4 - N_5$ and $N_1 - N_3 - N_5$. We let N_3 be the attacker that seeks to consistently alter the clock skew between N_1 and N_5. Initially, N_1's transmission reaches N_3, N_3 delays the transmission by some amount of time and then relays it to N_5. The reverse link from N_5 to N_1 via N_3 is set up with identical parameter values. N_3 then seeks to alter the skew by reducing the delay of the $N_1 - N_3 - N_5$ link while enlarging the delay of the $N_5 - N_3 - N_1$ link at a constant rate over time. Eventually the attacker cannot reduce the delay further and the measured skew at that instance would differ significantly from the expected measured skew. The time of arrival would thus also differ significantly from the ETA. When the next packet arrives, the measured skew is updated with exponential smoothing; thus, we should observe a exponential fall-off in the ETA error.

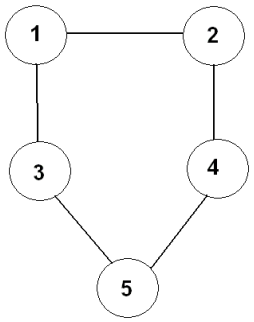
In our implementation, N_1 sends a packet to N_5 every 3 s. We experimented with two scenarios. In the first scenario, we let the attacker delay packets for 10000 μs on both $N_1 - N_3 - N_5$ and $N_5 - N_3 - N_1$ links initially. We then measure the error in the ETA of packets while altering the constant rate of change in the attacker's delays from 10 μs per packet to 160 μs per packet. The error in ETA over time is shown in Figure 4(b). The attacker that alters the measured skew by reducing delays 160 μs every packet, who also alters the measured skew the most among the set of attackers, is the first attacker that cannot further reduce the delay. Since this attacker altered the measured skew the most, the error in ETA is also the largest.

We performed another experiment where the initial delay is 1000 μs, and attackers alter the skew by reducing it at rates from 1 μs per packet to 5 μs per packet. The results of the second experiment are shown in Figure 4(c). The spike in the error of the ETA is much smaller compared to the first experiment. In particular, an attacker that alters the skew by reducing the delay by 1 μs per packet does not create a significant spike in the error in ETA due to noise.

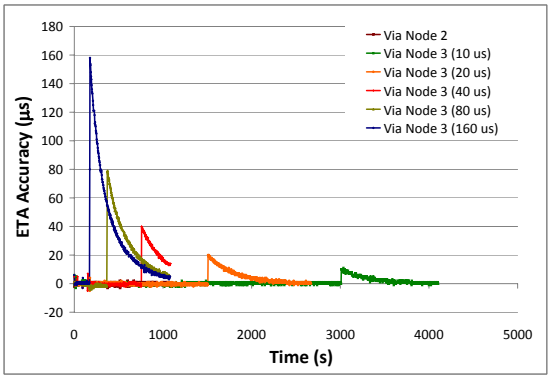
C. Evaluation Environment of the Network-wide Clock Synchronization

We implemented our protocol on top of the implementation evaluated in Section V-A. We tested our network-wide implementation on a 25 node network.

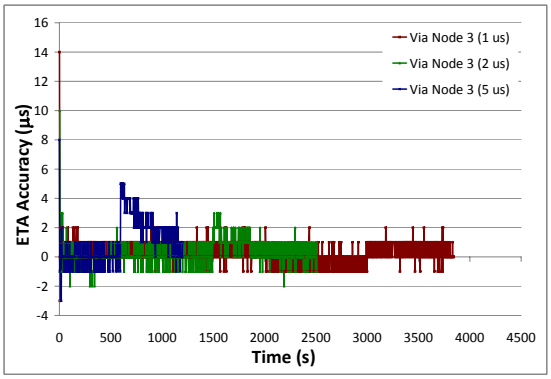
We ran several experiments; in some scenarios, we did not introduce any attackers, whereas in others, we introduced attackers that result in inconsistent link behavior. We used two general scenarios to induce the nodes to think there is an attacker, (i) making two nodes that share a link exchange invalid skews between them, or (ii) making one of the neighbors of a given node A advertise an incorrect skew, which results in non-compliances with respect to node A when other nodes in the network update their reachable table. In the first case, when a compromised link is detected, legitimate nodes stop using it, and since it not advertised, the link is excluded from the network. In the second case, since there are non-compliances in the reachable lists of the nodes with respect to node A, a node wishing to communicate securely with A must first remove all non-compliant skews. All experiments, regardless of the scenario used, showed similar results: the product of the skews between nodes



(a) 5-Node Network Topology



(b) Error in ETA with initial delay of 10000 μs



(c) Error in ETA with initial delay of 1000 μs

Fig. 4. Error in ETA over time

in the network stayed always very close to 1, even over multi-hop links. Furthermore, when a compromised link is detected, nodes stop using such link either because it is never advertised, or because it is eliminated as a result of the last step of the protocol (removal of non-compliant skews).

VI. Conclusion

We have presented a synchronization scheme that provides network-wide clock synchronization. We constructed our protocol on top of the scheme recently proposed by [1], extending it to provide network-wide clock synchronization, topology discovery, and elimination of inconsistent attackers.

Moreover, our protocol examines every path of interest to eliminate any wormhole attackers that are consistent on a per-link basis but not on a global basis. We disseminate link information network-wide, allowing each node in the network to learn the topology of the network and to remove inconsistent links from the network.

We have implemented our protocol on IMote2 sensors and demonstrated that it can provide sub-microsecond accuracy for estimating the time of arrival of a packet, and that we can also very accurately measure skew, since the measured product-of-skews is very close to one. Furthermore, we show that inconsistencies are eliminated from the system by removing links that are in control of an inconsistent attacker.

References

[1] J. Chiang, J. Haas, Y.-C. Hu, P. Kumar, and J. Choi, "Fundamental limits on secure clock synchronization and detection of man-in-the-middle attacks," *INFOCOM 2009*, April 2009.

[2] Y.-C. Hu, A. Perrig, and D. Johnson, "Wormhole attacks in wireless networks," *Selected Areas in Communications, IEEE Journal on*, vol. 24, no. 2, pp. 370–380, Feb. 2006.

[3] S. Graham and P. Kumar, "Time in general-purpose control systems: the control time protocol and an experimental evaluation," *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 4, pp. 4004–4009 Vol.4, Dec. 2004.

[4] N. Freris and P. Kumar, "Fundamental limits on synchronization of affine clocks in networks," *Decision and Control, 2007 46th IEEE Conference on*, pp. 921–926, Dec. 2007.

[5] Crossbow Technology, "IMote2 IPR2400 datasheet." [Online]. Available: http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/Imote2_Datasheet.pdf

[6] "TinyOS." [Online]. Available: http://www.tinyos.net/

[7] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 147–163, 2002.

[8] R. Solis, V. Borkar, and P. Kumar, "A new distributed time synchronization protocol for multihop wireless networks," *45th IEEE Conference on Decision and Control*, pp. 2734–2739, Dec. 2006.

[9] A. Giridhar and P. Kumar, "Distributed clock synchronization over wireless networks: Algorithms and analysis," in *Decision and Control, 2006 45th IEEE Conference on*, Dec. 2006, pp. 4915–4920.

[10] R. Karp, J. Elson, D. Estrin, and S. Shenker, "Optimal and global time synchronization in sensornets," UCLA, Tech. Rep., 2003.

[11] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *SenSys 2003*. New York: ACM Press, Nov. 2003, pp. 138–149.

[12] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi, "The flooding time synchronization protocol," in *SenSys 2004, Baltimore, MD, USA, Nov. 3-5, 2004*, J. A. Stankovic, A. Arora, and R. Govindan, Eds. ACM, 2004, pp. 39–49.

[13] W. Su and I. F. Akyildiz, "Time-diffusion synchronization protocol for wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 13, no. 2, pp. 384–397, 2005.

[14] Q. Li and D. Rus, "Global clock synchronization in sensor networks," *Computers, IEEE Transactions on*, vol. 55, no. 2, pp. 214–226, Feb. 2006.

[15] M. Manzo, T. Roosta, and S. Sastry, "Time synchronization attacks in sensor networks," in *SASN '05: Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks*. New York, NY, USA: ACM, 2005, pp. 107–116.

[16] A. Boukerche and D. Turgut, "Secure time synchronization protocols for wireless sensor networks," *Wireless Communications, IEEE*, vol. 14, no. 5, pp. 64–69, October 2007.

[17] S. Ganeriwal, S. Čapkun, C.-C. Han, and M. B. Srivastava, "Secure time synchronization service for sensor networks," in *WiSe '05*. New York, NY, USA: ACM, 2005, pp. 97–106.

[18] K. Sun, P. Ning, and C. Wang, "Secure and resilient clock synchronization in wireless sensor networks," *Selected Areas in Communications, IEEE Journal on*, vol. 24, no. 2, pp. 395–408, Feb. 2006.

[19] D. Bertsekas and R. Gallager, *Data networks*. Englewood Cliffs, New Jersey: Prentice-Hall, 1987.